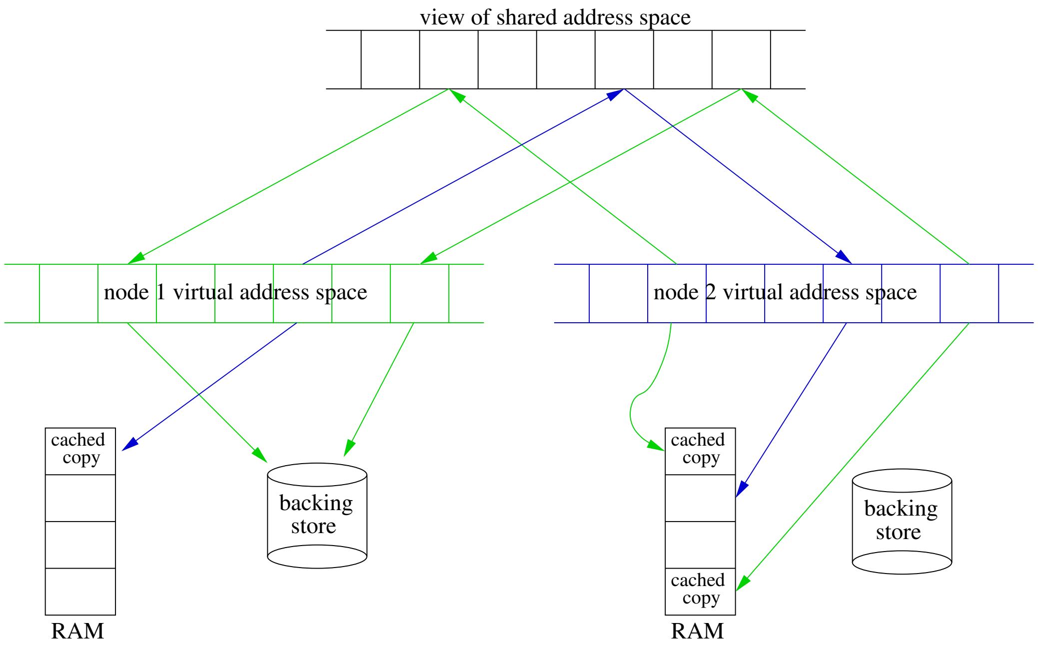
Distributed Mungi

Daniel Potts

danielp@cse.unsw.edu.au

University of New South Wales, Sydney, Australia

Distributed shared memory overview

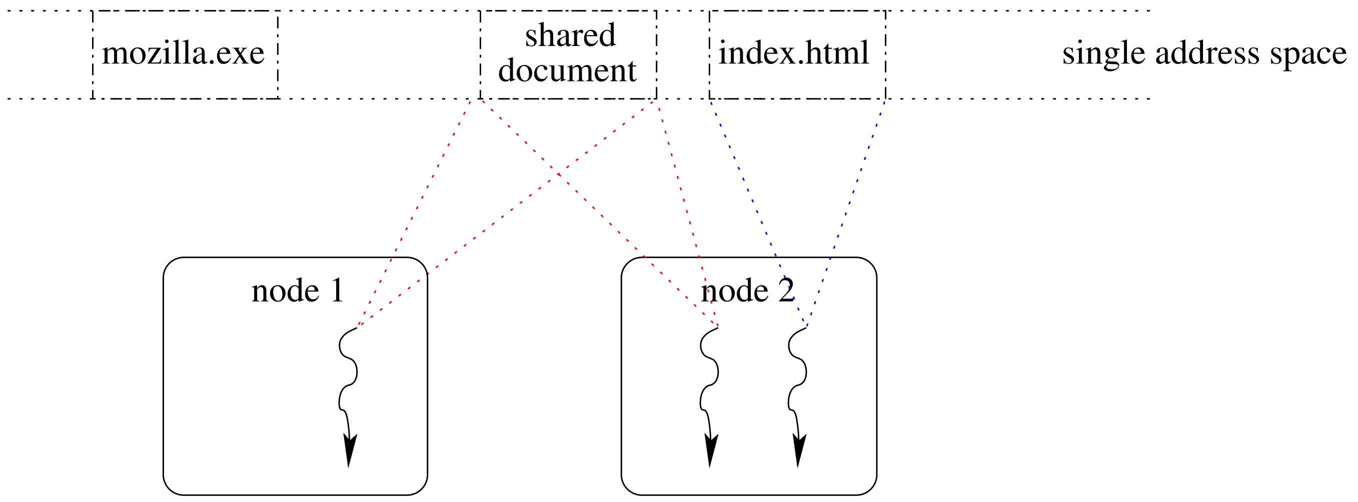


Typical DSM:

- Nodes see same data in a shared address space region.
- Page table tracks mappings.
- Meta-state stored to maintain data coherency.

Distributed SAS overview

A SAS provides an excellent platform for transparently distributing directly addressable data.



- Simplifies communication.
- Location independent.

The SAS is huge.

- 1 - 1,000,000+ nodes ?
- densely populated with over 17 trillion objects ?
- objects located and accessed sporadically.
- non-uniform, geographically separated, network topology ?

SAS distribution problem [2]

Shared memory data and consistency state must be

- quickly located with a minimum amount of communication overhead,
- efficiently transfered,

in a *scalable* and performance oriented manner.

Model

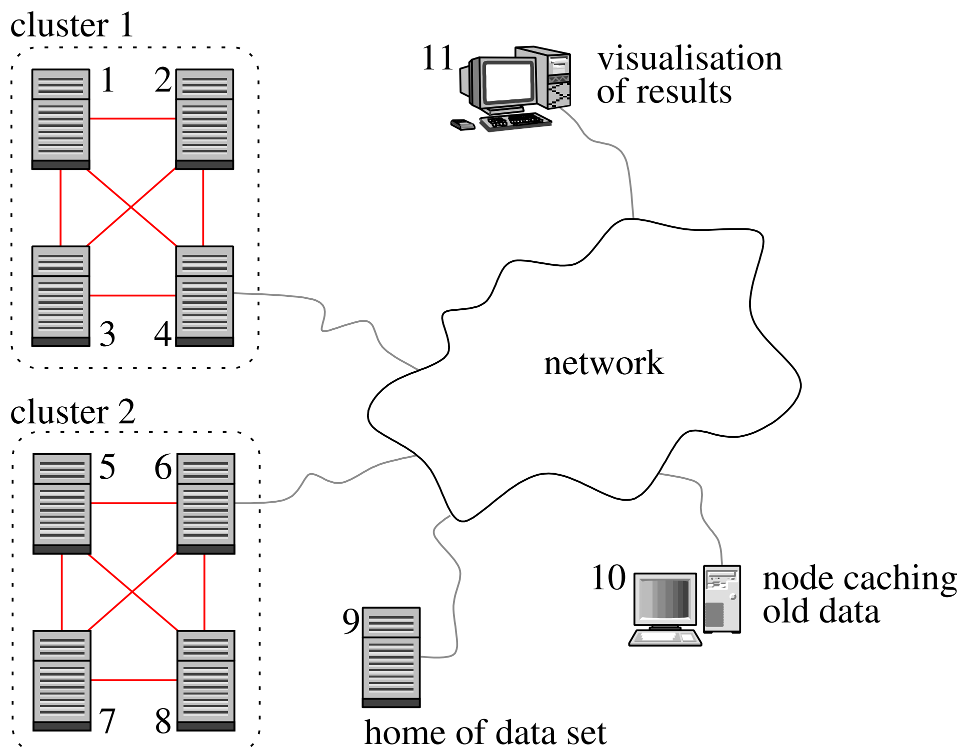
Focus on minimising overhead for

- communication,
- memory usage,
- consistency meta-state related computation.

This is done by providing consistency domains that

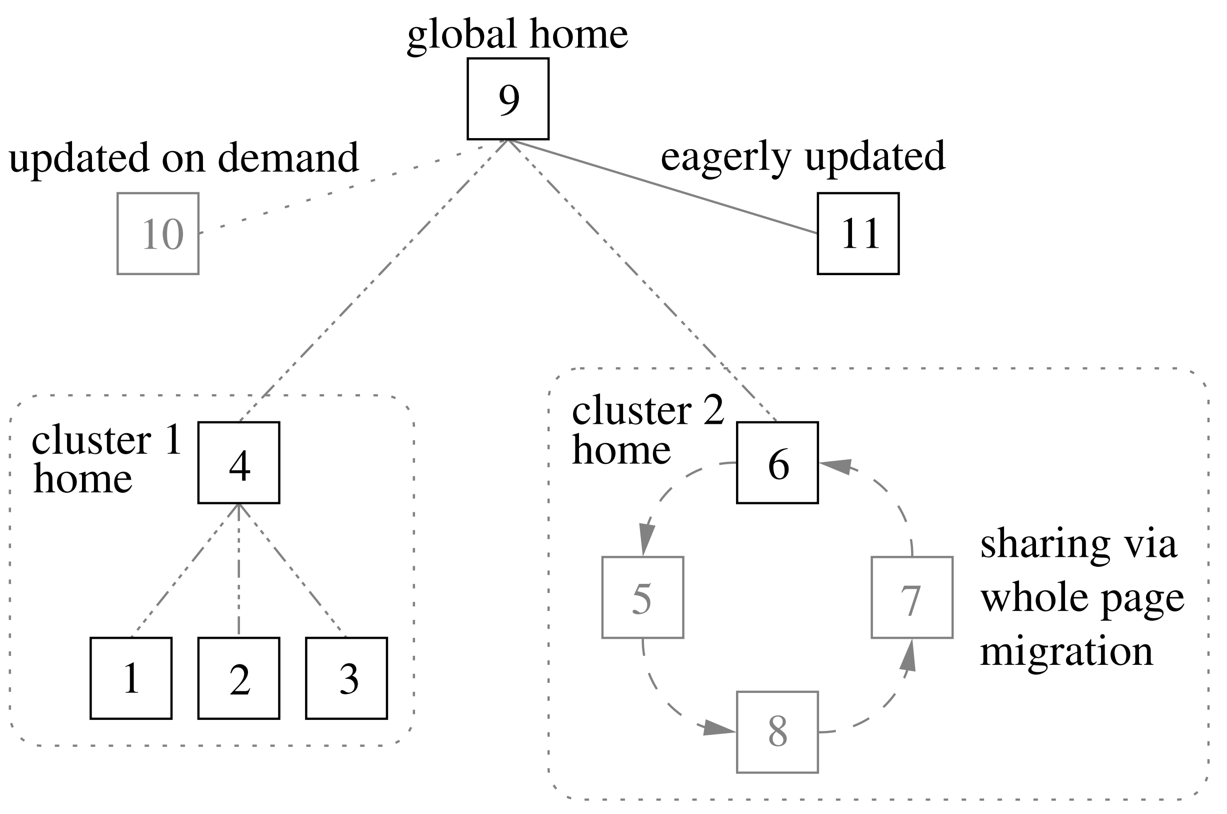
- allow multiple levels of consistency for both threads and objects,
- restrict the flow of and translate meta-state information,
- allow different update protocols to be used within a domain,
- allow for relaxed visibility by separation into separate domains.

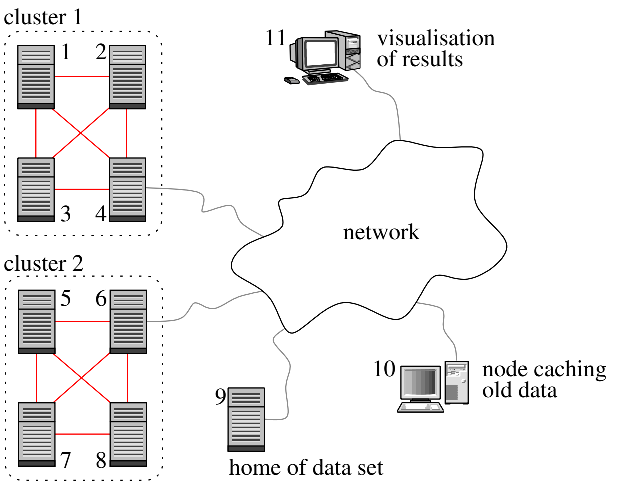
Example system:



- multiple sharers accessing the same data,
- different interconnects, geographic and administrative domains.

Mungi cluster example [2]



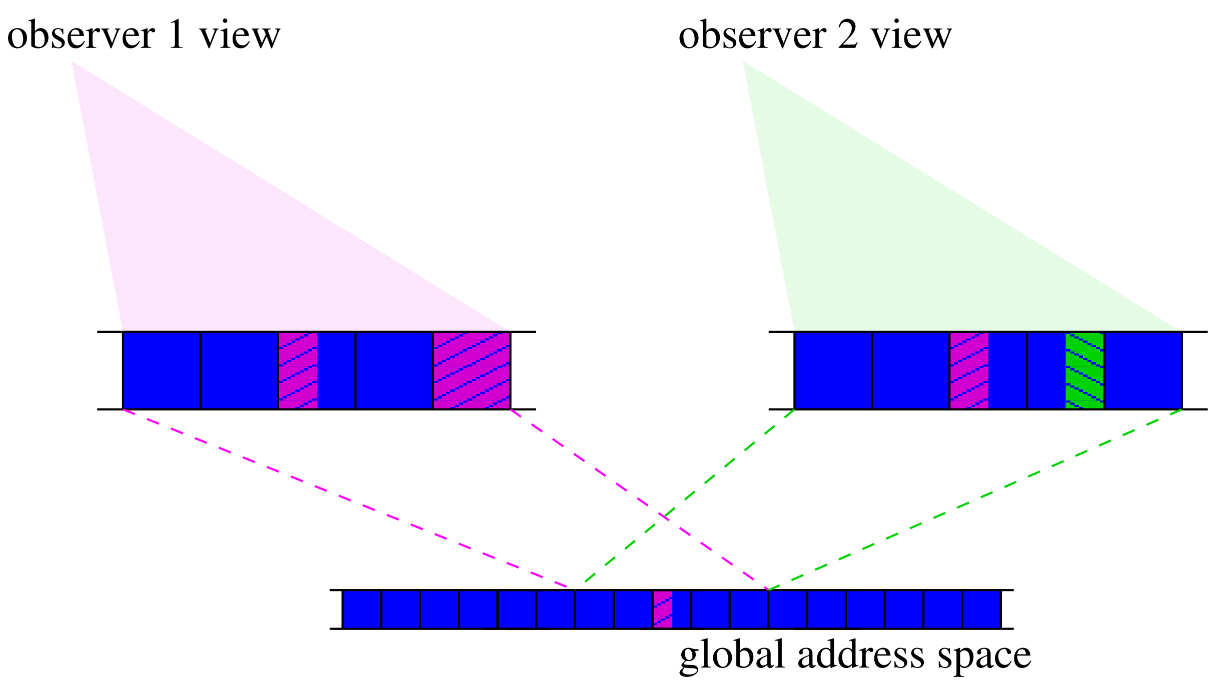


- fast communication within clusters,
- updates propagate to *global home* as needed,
- node 11 sees interactive changes without impacting on computation.

Visibility

To reduce communication:

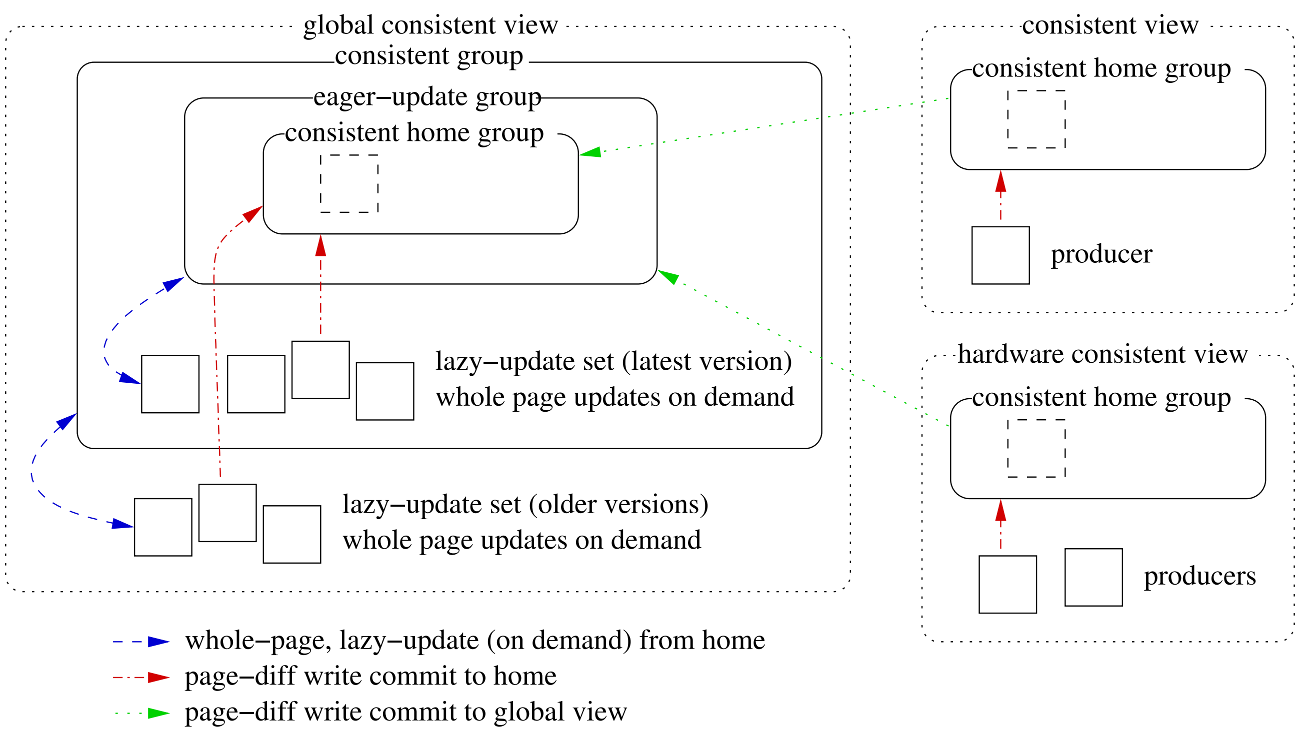
- exploit laziness of visibility,
- choose appropriate update method.



Visibility

To reduce communication:

- exploit laziness of visibility,
- choose appropriate update method.



Meta-state:

- Meta-state assists in reducing communication (and latency).
- provides information on what updates are needed.

Examples include

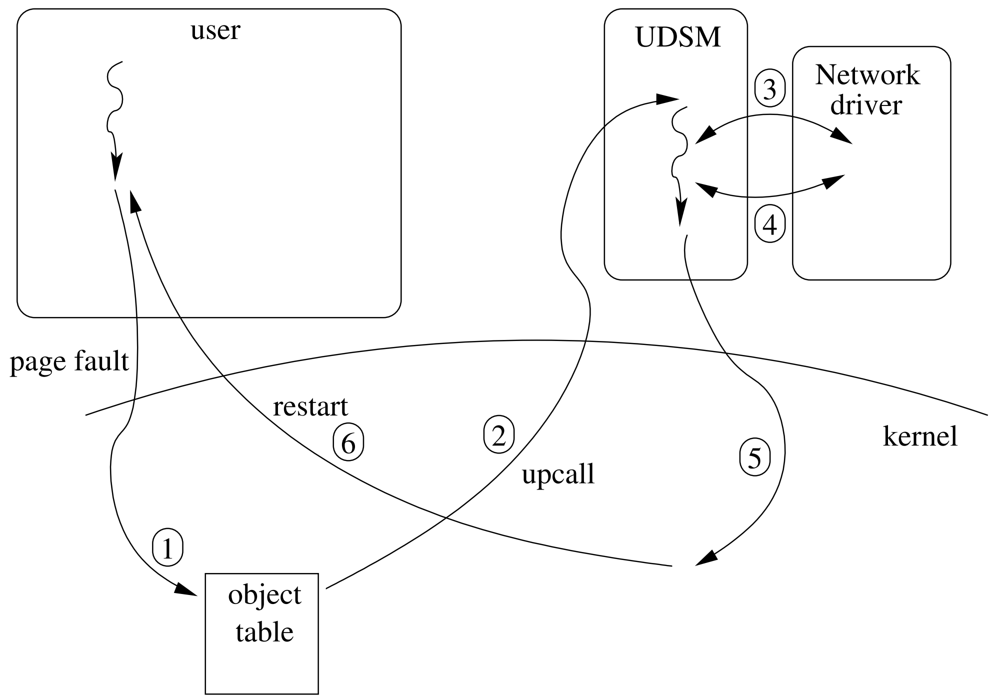
- Vector time-stamps,
- Versioned page dependencies
- Function dependencies.

Mungi DSM components:

- kernel paging interface,
- user-level DSM pager,
    - shared memory data transfer,
    - consistency state maintenance,
    - data location overlay,
- low-latency, stateless communication..

... some of these we go into detail.

Mungi DSM Internals



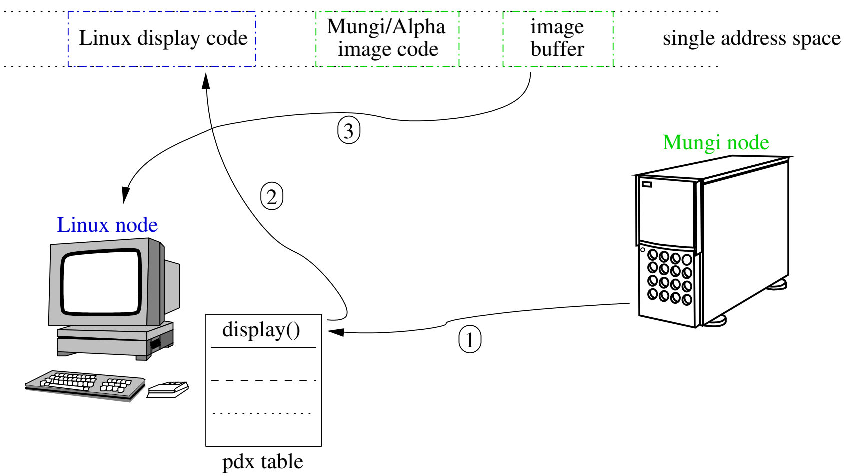
1. Kernel looks up fault region and determines it is non-local.
2. Kernel executes PDX upcall to registered user-level DSM handler.
3. User-level handler locates remote region owner.
4. User-level handler requests page data from region owner.
5. All data is received, PDX upcall returns to kernel.
6. Kernel restarts faulting thread.

What good is it in Mungi?

- DSM gives us network transparency,
- remote-PDX gives us ability to invoke remote procedures.

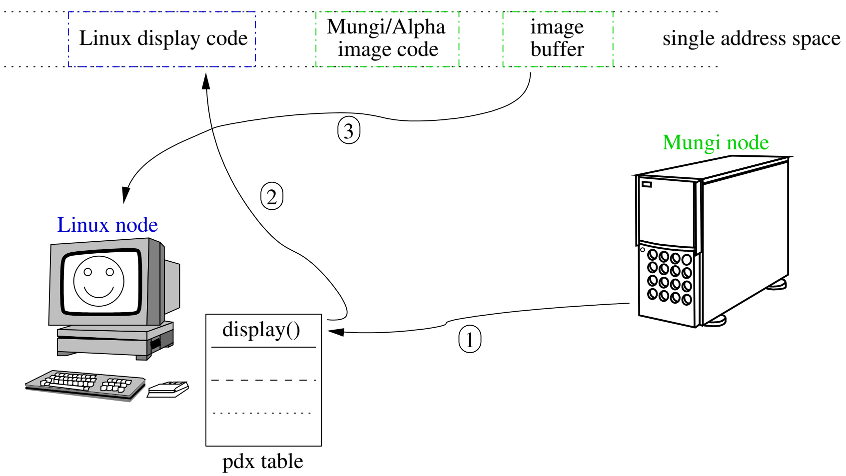
We can do many cool things..

Remote PDX example



1. Mungi node issues remote pdx call
2. Linux node verifies and executes entry point
3. Data is paged across into frame buffer

Remote PDX example



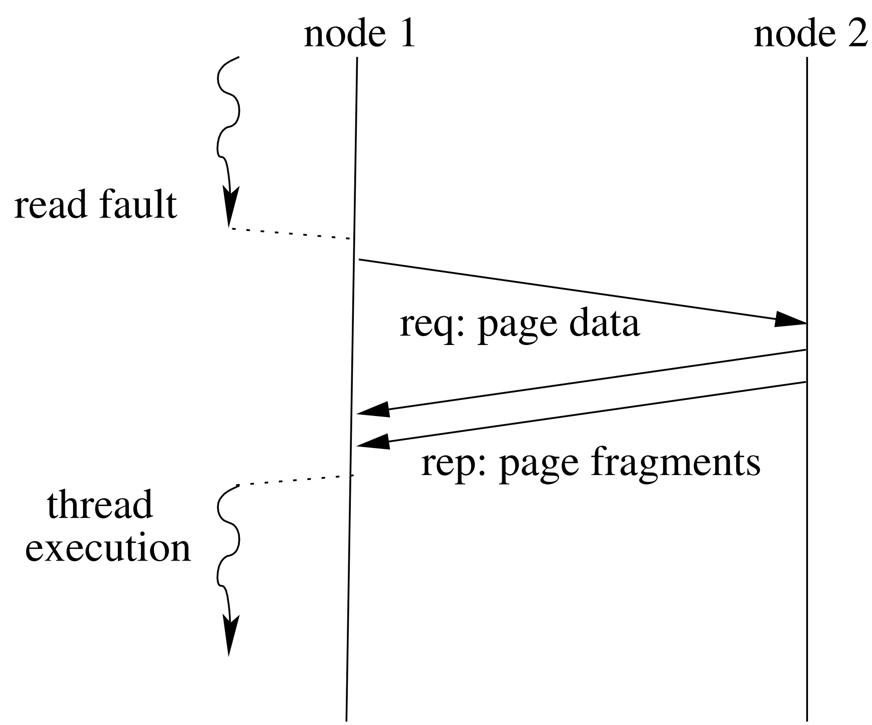
1. Mungi node issues remote pdx call
2. Linux node verifies and executes entry point
3. Data is paged across into frame buffer

DSM communication library

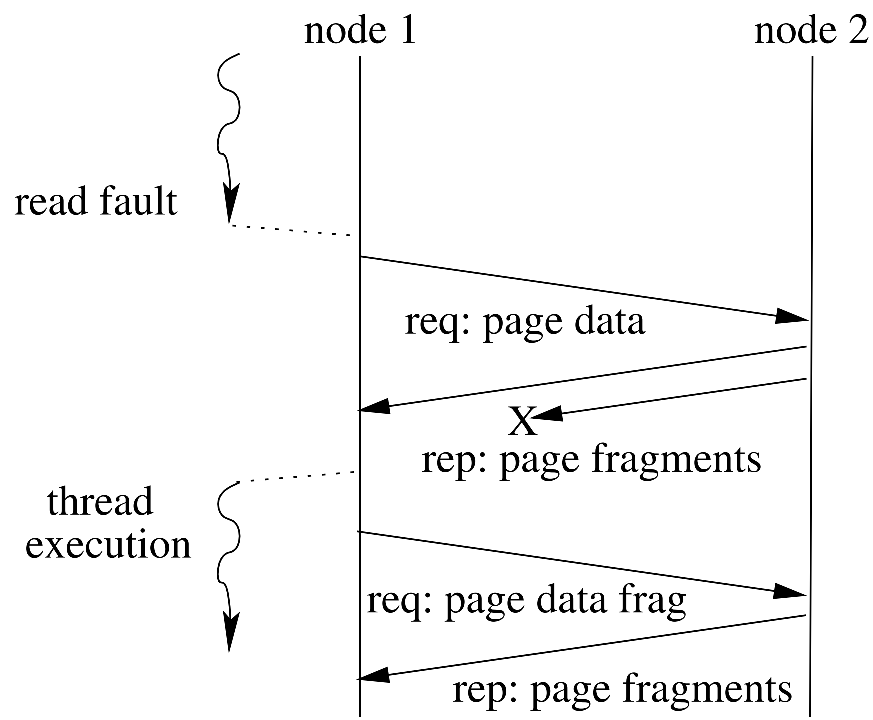
A low latency, high performance communication library.

- designed to be stateless,
- currently runs on raw Ethernet,
- several data transfer methods (whole region, diff)
- region owner discovery protocol
- remote pdx protocol

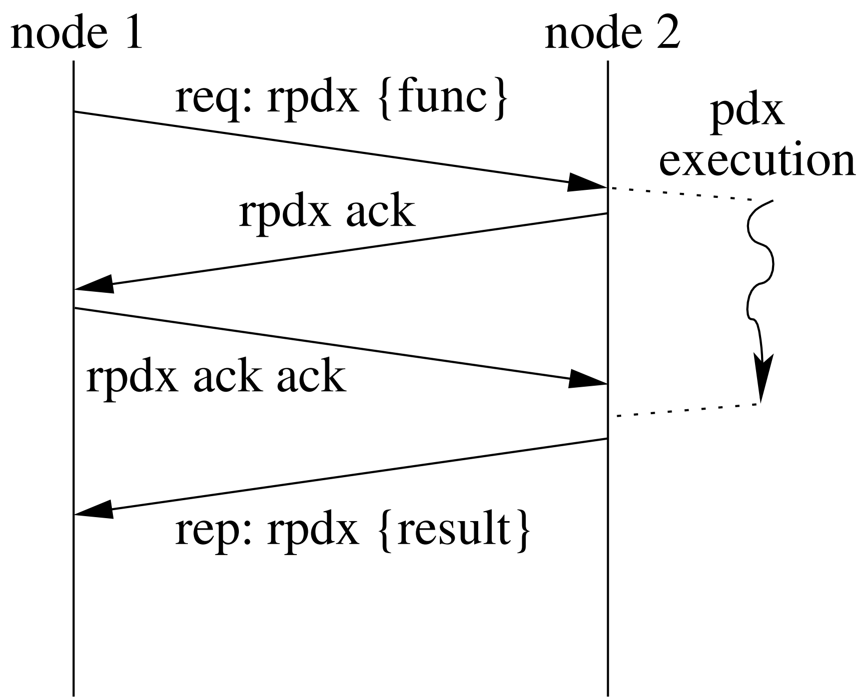
Page fault on remote memory example



Page fault on remote memory example



Remote PDX example



On-going work

- Scalable overlay network that maps data address to location.
- Implement various data transfer methods.
- Implement communication over TCP/IP.
- Scalable, version tagged, consistency meta-state.
- Measure performance characteristics of communication, paging system.
- Programming models.